Chat Bot through Azure Luis

1. **Introduction :**

   A chatbot is an artificial intelligence (AI) software that can simulate a conversation (or a chat) with a user in natural language through messaging applications, websites, mobile apps or through the telephone.

   1.1 Importance of ChatBot:

   Chatbot is often described as one of the most advanced and promising expressions of interaction between humans and machines. However, from a technological point of view, a chatbot only represents the natural evolution of a Question Answering system leveraging Natural Language Processing (NLP).

   1.2 Applications of ChatBot:

   - Pizza Order.
   - Product Suggestion
   - Weather Report
   - News and etc.,

2. **Prerequisites:**
   - Python – Version : 3.

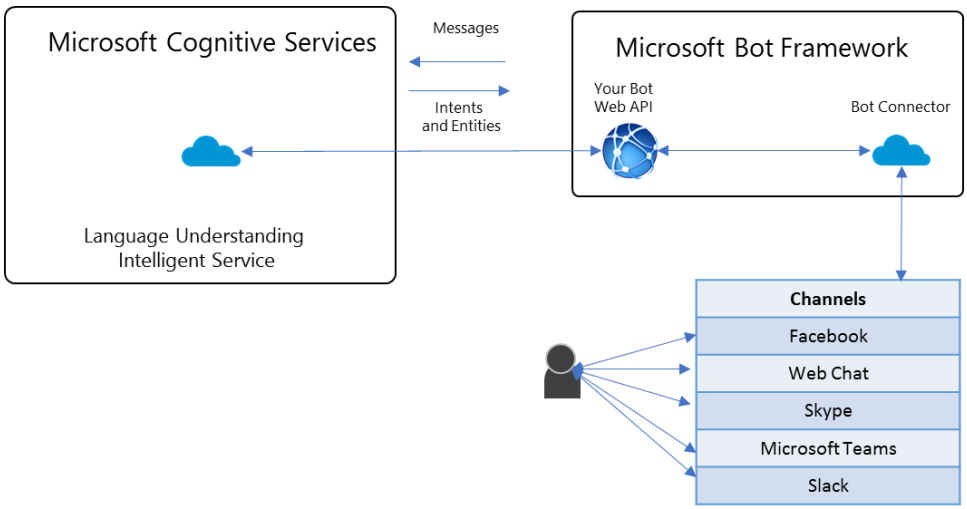
     Azure Luis

3. **Introduction to Google DialogFlow:**

   3.1 About Google DialogFlow:

   Azure LUIS: Language Understanding (LUIS) allows your application to understand what a person wants in their own words. Azure LUIS uses machine learning to allow developers to build applications that can receive user input in natural language and extract meaning from it.

   3.2 Architecture of DialogFlow:



4. **Problem Statement:**

4.1 About:

The aim is to build a chat bot which answers the user queries about the number of corona virus cases within Indian states and all countries across the globe with its geographic location, about the corona virus, the symptoms and the preventive measure.

5. **Solution – Approach:**

5.1 Technical Stack:

- Python – version 3.6
- Azure Luis
- Microsoft Bot Connector
- Microsoft cosmosDB – MongoAPI

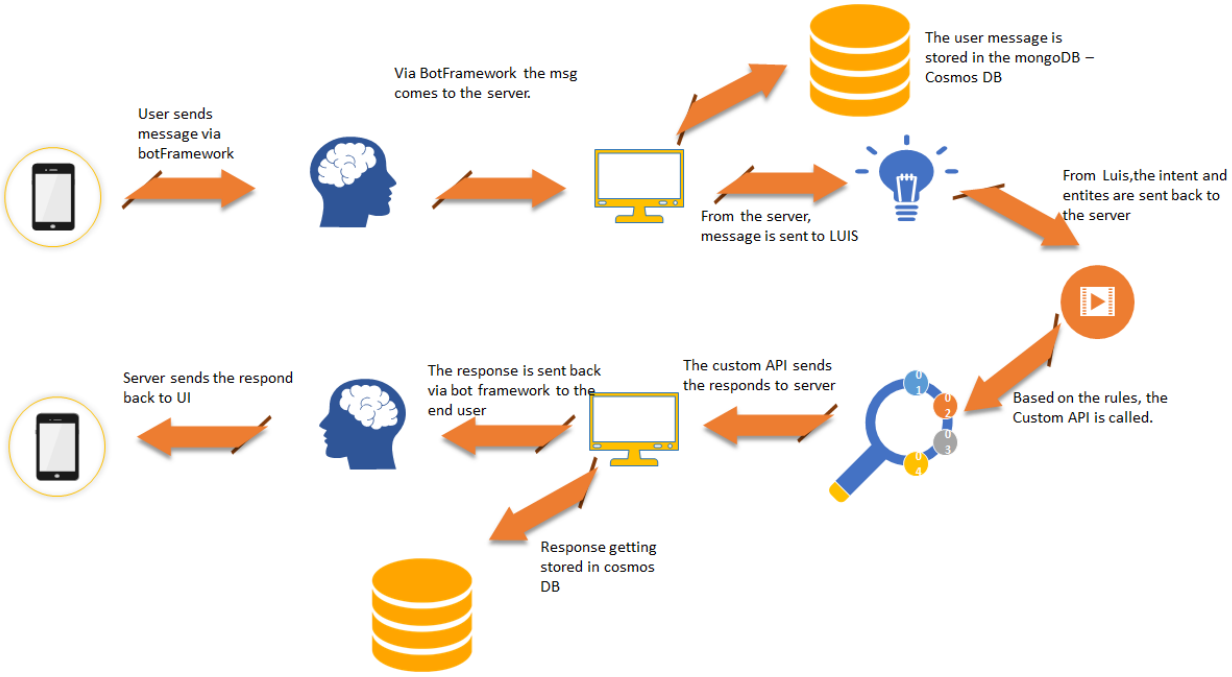
5.2 Libraries & Packages:

- Flask
- Pandas
- Requests
- Geopy

5.3 Third Party API's and License:

- API's used : https://corona.lmao.ninja/docs/
- Github : https://github.com/NovelCOVID/API

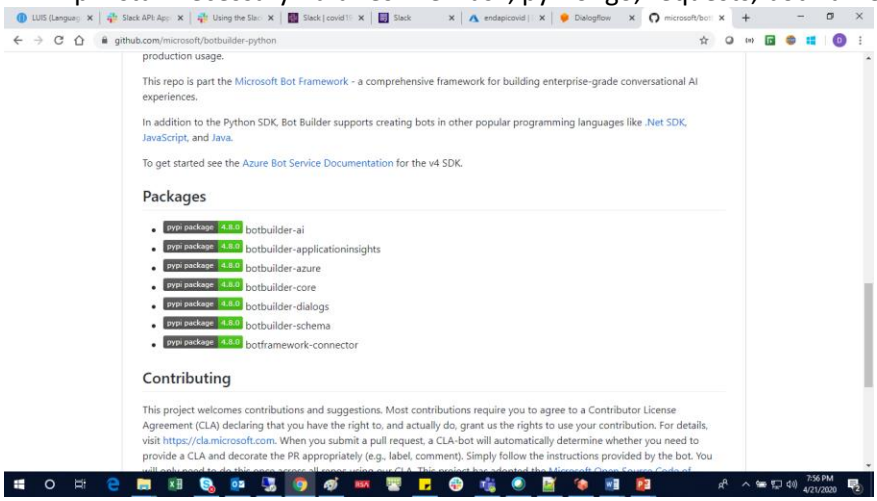
5.4 Solution Architecture:



6. Implementation :

6.1 Library installation :

- Create a conda environment using “conda create –n <env_name>”
- Activate the conda environment.
- Pip install necessary libraries like Flask, pymongo, requests, botFramework connectors.

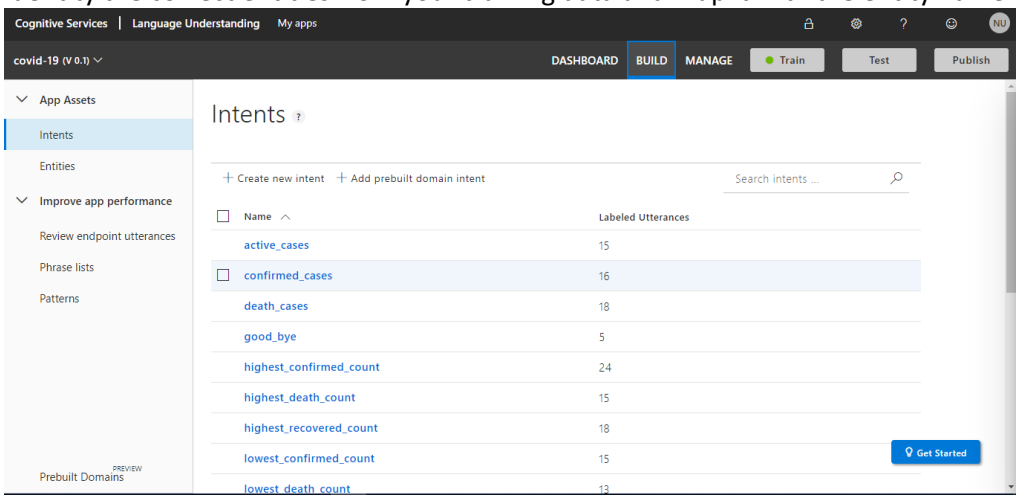


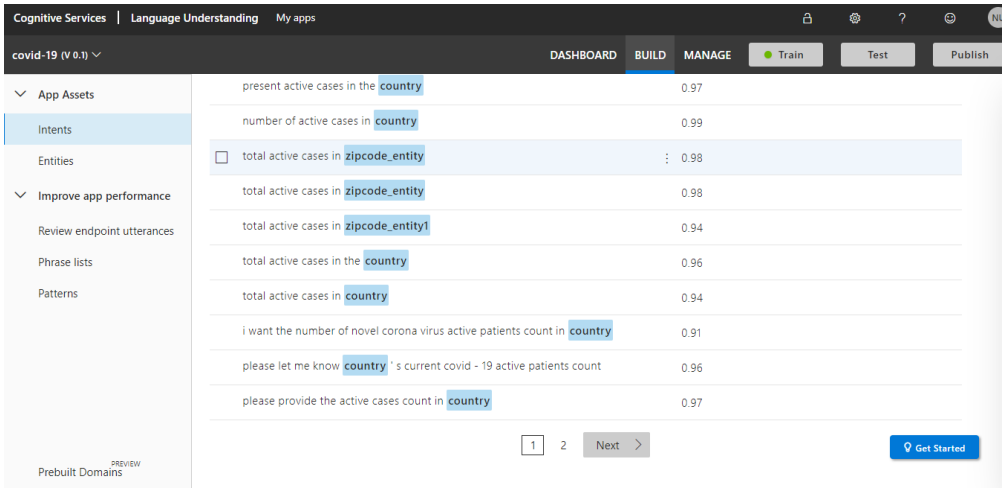
6.2 Project Setup:

- Create a project “azure_chatbot”, using pycharm.
- Create app.py where the Flask based REST api’s will be created.

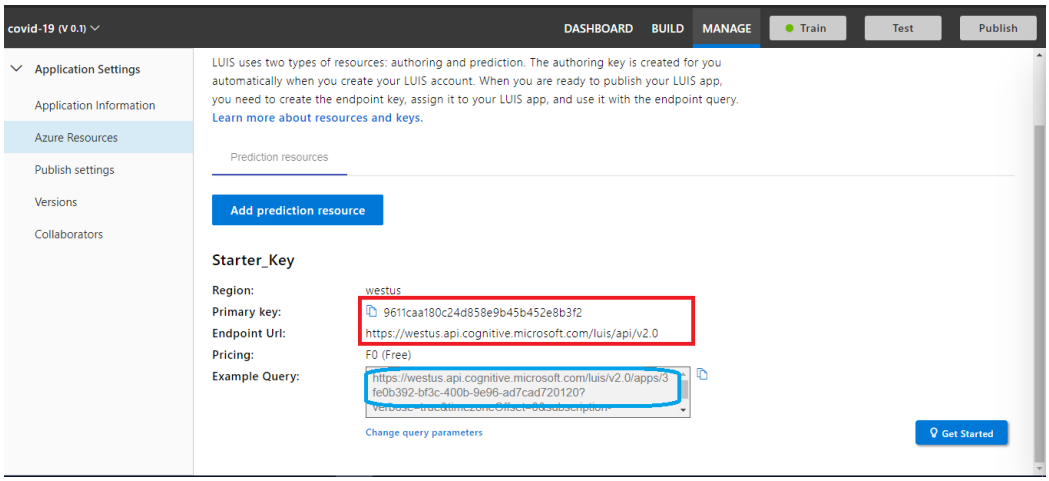
6.3 Training Data creation in AzureLuis :

- Create a new Free account in azure luis.
- Create a new App.
- From the Build option , start create new intent and give the intent name and start adding the training pharases.
- Identity the correct entities from your training data and map it with the entity name.



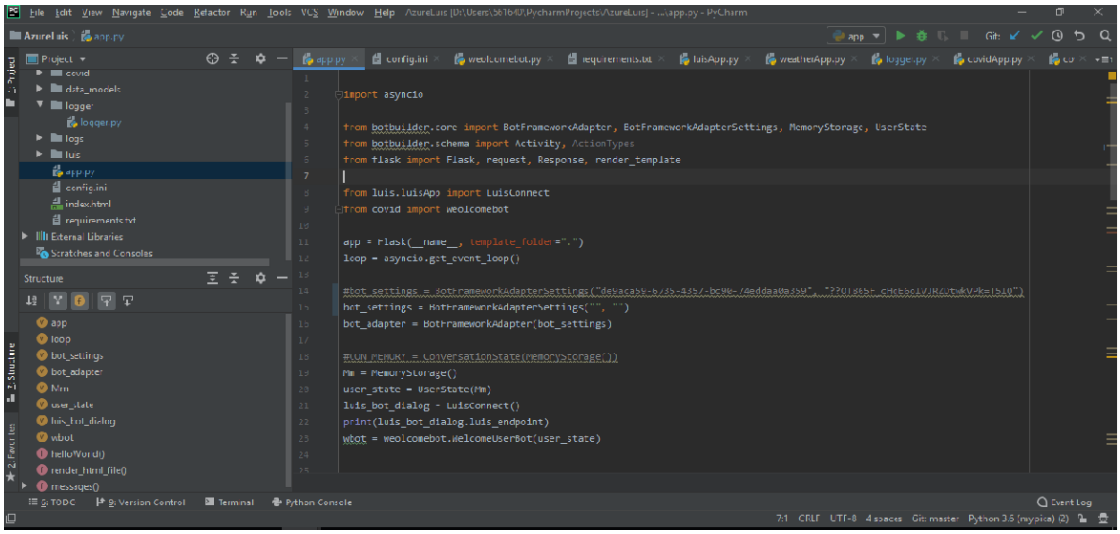


- Once the training data is created, train the complete app and test the same to check your results. Once satisfied publish the model to production.
- Go to Manage -> Azure Resources and the primary Key, endpoint url and the app id.

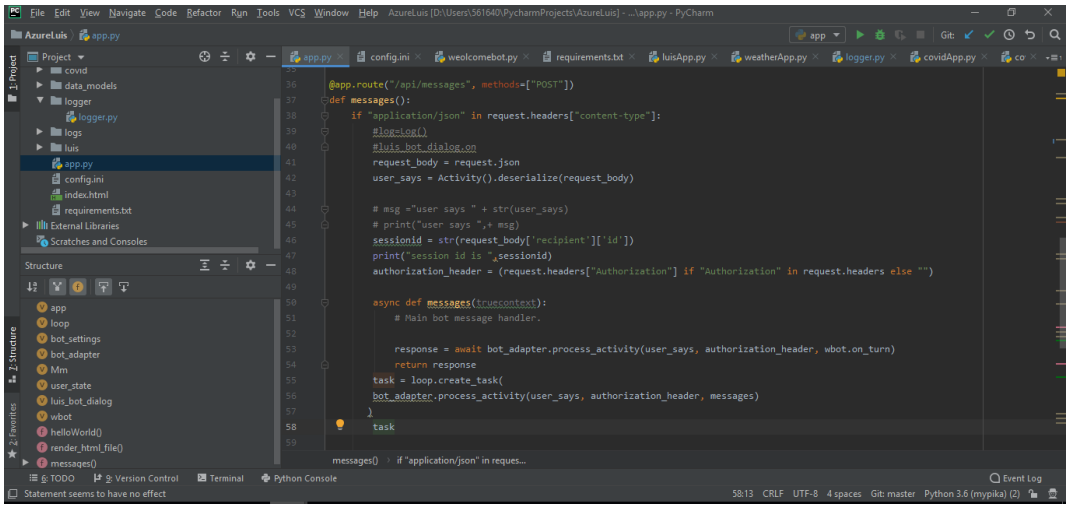


6.5 Python Project Setup:

- Create a new python project in pycharm.
- Create config.ini file and have the primary Key, app id and the endpoint url.
- Create a Microsoft bot luis bot connection.



- Create app.py file and create end pint “api/messages” to make connections to user and luis api.



- Builder your custom Rules engine to make the endpoint call and get the response back and send via the botframework to the end user.

6.4 Deploy the App to Azure:

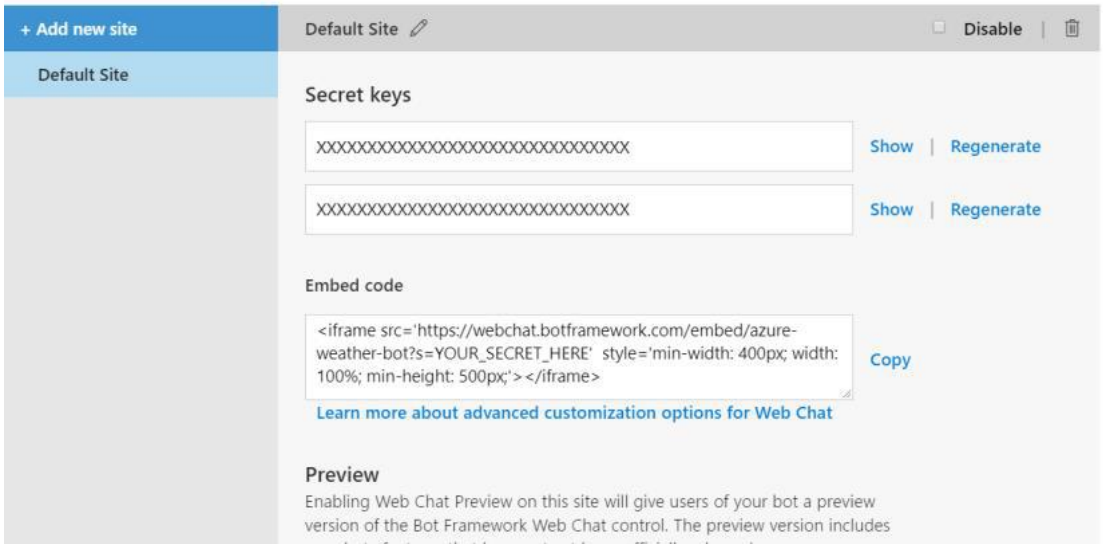
- Create the Azure Acocount.
- In the search, type web app, and select web App.
- Provide the app name, resource group(create new if necessary), runtime stack(node 10 LTS), region, select the 1 GB size, which is free to use. Click *Review+create* to create the web app.
- Once the deployment is completed, open the app and go to the 'Deployment Center' option. Select 'local git' for source control and click continue.
- Select the kudo 'App service build provider' as the build provider and click continue and Click Finish.
- Go to the overview section of the app, and the Git link now will be visible.
- Copy the git link and got the pycharm, under VCS enable version control with GIT.
- Select your project Goto GIT -> Repository -> Remote Add the git link.
- Select your project, GIT Add.
- Select your Project, GIT commit directory and push. Get the git credentials from the deployment center.
- After successful push, the web app will be deployed.

6.5 Create bot Channel Registration:

- Now in the Azure portal, create a bot channel registration.
- Provide the bot handle, resource group and other fields.
- For Message endpoint, provide: <URL from the web app created above>api/messages. Click Create to create the bot.
- Once your web channel registration gets done, open the bot and then click 'test in web chat.' If the chat works fine, our deployment is a success.

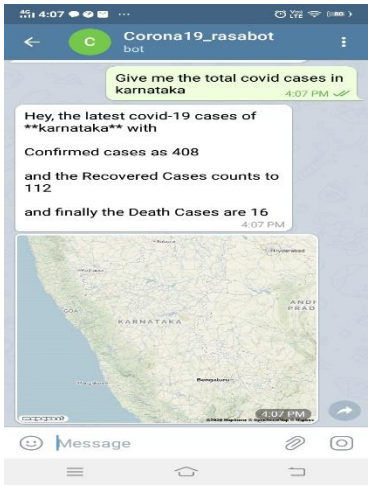
6.6 WEB/HTML Embed deployment:

- Go to the channels section of your bot.
- The bot can be deployed as an embedding to an existing HTML page by selecting the get bot embedded code option



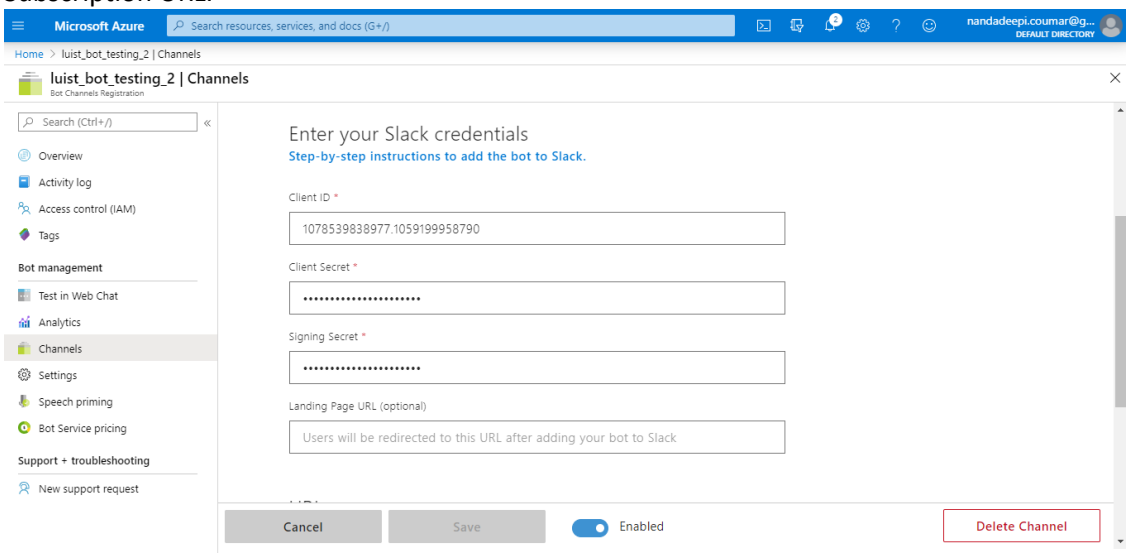
6.7 Telegram Integration:

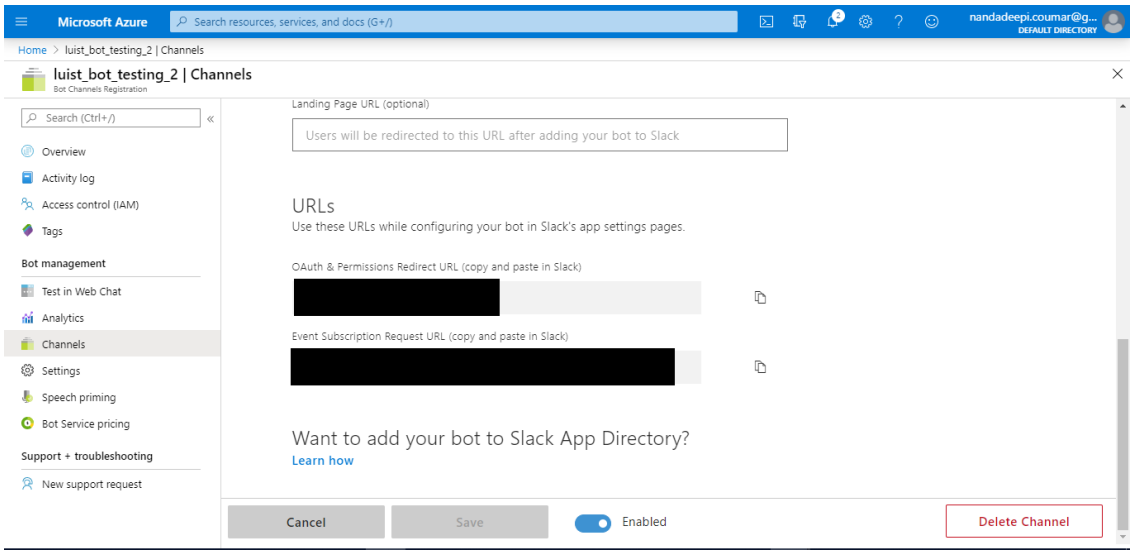
- Open the telegram app and search for botfather(it is an inbuilt bot used to create other bots)
- Start a conversation with botfather and enter /newbot to create a newbot.
- Give a name to your bot
- Give a username to your bot, which must end in _bot.
- This generates an access token. Enter that access token after clicking the telegram channel in your bot app and click save.



6.8 Slack Integration:

- For Slack integration, we need to create a slack app first (https.//api.slack.com/apps)
- After creating the slack app, from basic information section copy, the client ID, client secret, signing secret
- From the Microsoft azure bot channel, under Channels Tab, select Slack paste the client ID, client secret and signing secret and generate the OAuth & Permission URL and the Event Subscription URL.





- From the generated URL, copy the Event Subscription URL and go the Slack APP, under the Events & Subscriptions, Give the Redirect URL.
- Go the Subscribe Bot Events, select the necessary event like message:mentions, message:read, channel:read.
- Copy the generated OAuth URL and in the slack app under the OAuth & permission give the redirect URL and click Save.
- Configure the necessary Scopes under OAuth & Permissions.

- Go to Slack and select your App Service and start your chat.